Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method of handling an exception, the method comprising:

recognizing that [[an]] the exception has occurred during [[the]] execution of a given method;

consulting a stack frame associated with [[said]] the given method to determine an identity of an object required to be unlocked; and

removing [[said]] a lock on [[said]] the object.

2. (Currently Amended) The method of claim 1 further comprising:

determining a program counter address at which [[said]] the exception has occurred;

determining, from a table, a synchronization depth associated with [[said]] the program counter address; and

wherein [[said]] the consulting [[said]] the stack frame includes reading from a location associated with [[said]] the synchronization depth.

3. (Currently Amended) A computer program product stored in a computer readable medium, wherein the computer program product comprises computer-executable instructions that compose an [[An]] exception handler, wherein the exception handler is operable to:

recognize that an exception has occurred during [[the]] execution of a given method;

consult a stack frame associated with [[said]] the given method to determine an identity of an object required to be unlocked; and

remove [[said]] a lock on [[said]] the object.

4. (Currently Amended) A computer readable medium containing computer-executable instructions which, when performed by a processor in a computer system, cause [[said]] the computer system to:

recognize that an exception has occurred during [[the]] execution of a given method;

consult a stack frame associated with [[said]] the given method to determine an identity of an object required to be unlocked; and

remove [[said]] a lock on [[said]] the object.

5. (Currently Amended) A method of generating executable code at [[At]] a just in time compiler of programming language code, [[said]] wherein the programming language code ~~including~~ includes a plurality of instructions, ~~a method of generating executable code~~ the method comprising:

determining a synchronization depth for [[said]] each instruction in the plurality of instructions;

associating [[said]] the synchronization depth with a program counter address associated with [[said]] each instruction in the plurality of instructions;

determining a continuous range of program counter addresses associated with an equivalent synchronization depth; and

storing an indication of [[said]] the continuous range of program counter addresses in a table associated with [[said]] the equivalent synchronization depth.

6. (Currently Amended) The method of claim 5 wherein [[said]] the programming language code is Java.

7. (Currently Amended) The method of claim 5, wherein a first synchronization depth determined for a first instruction differs from a second synchronization depth determined for a directly preceding instruction, wherein determining [[said]] the continuous range of program counter addresses comprises~~,~~ ~~where a first synchronization depth determined for a first instruction differs from a second synchronization depth determined for a directly preceding instruction,~~ storing [[said]] the second synchronization depth in [[said]] the table, wherein the table is associated with a given range of program counter addresses, ~~where said~~ and wherein the given range of program counter addresses includes [[said]] a program counter address of [[said]] a directly preceding instruction.

8. (Currently Amended) The method of claim 5, ~~where said~~ wherein the plurality of instructions may be arranged in a plurality of basic blocks of code, and wherein the method further comprises: ~~further comprising~~

determining a synchronization depth at [[the]] a beginning of each [[said]] basic block of code in the plurality of basic blocks of code.

9. (Currently Amended) A just in time compiler of programming language code, [[said]] wherein the just in time compiler is stored in a computer-readable medium, wherein the programming language code ~~including~~ includes a plurality of instructions, [[said]] and wherein the compiler is operable to:

determine a synchronization depth for [[said]] each instruction in the plurality of instructions;

associate [[said]] the synchronization depth with a program counter address associated with [[said]] each instruction in the plurality of instructions;

determine a continuous range of program counter addresses associated with an equivalent synchronization depth; and

store an indication of [[said]] the continuous range of program counter addresses in a table associated with [[said]] the equivalent synchronization depth.

10. (Currently Amended) A computer readable medium containing computer-executable instructions which, when performed by a processor in a computer system, cause [[said]] the computer system to:

determine a synchronization depth for [[said]] each instruction in a first plurality of instructions that compose a programming language code;

associate [[said]] the synchronization depth with a program counter address associated with [[said]] each instruction in the first plurality of instructions;

determine a continuous range of program counter addresses associated with an equivalent synchronization depth; and

store an indication of [[said]] the continuous range of program counter addresses in a table associated with [[said]] the equivalent synchronization depth.

11. (New) The method of claim 1 wherein consulting the stack frame comprises consulting only the stack frame.

12. (New) The method of claim 1 wherein a just in time compiler dedicates a storage location in the stack frame and stores, in the storage location, a reference to the object, and wherein consulting the stack frame comprises consulting the storage location for the reference.

13. (New) The method of claim 1 further comprising:

responsive to executing a syncenter bytecode in a JAVA virtual machine, placing a reference to the object into the stack frame.

14. (New) The method of claim 13 further comprising:

determining a storage location of the reference in the stack frame according to a depth of the syncenter bytecode in a tree of inlined invocations for the given method.

15. (New) The method of claim 13 further comprising:

releasing the lock by executing a syncexit bytecode in the JAVA virtual machine.

16. (New) The computer readable medium of claim 4 wherein consulting the stack frame comprises consulting only the stack frame.

17. (New) The computer readable medium of claim 4 wherein a just in time compiler dedicates a storage location in the stack frame and stores, in the storage location, a reference to the object, and wherein consulting the stack frame comprises consulting the storage location for the reference.

18. (New) The computer readable medium of claim 4 wherein the computer-executable instructions, when performed by a processor in a computer system, further cause the computer system to:

responsive to executing a syncenter bytecode in a JAVA virtual machine, place a reference to the object into the stack frame.

19. (New) The computer readable medium of claim 18 wherein the computer-executable instructions, when performed by a processor in a computer system, further cause the computer system to:

determine a storage location of the reference in the stack frame according to a depth of the syncenter bytecode in a tree of inlined invocations for the given method.

20. (New) The computer readable medium of claim 18 wherein the computer-executable instructions, when performed by a processor in a computer system, further cause the computer system to:

release the lock by executing a syncexit bytecode in the JAVA virtual machine.